2005

A Bipartite Graph Approach to Generate Optimal Test Sequences for Protocol Conformance Testing using the Wp-method

Jun Wang
Edith Cowan University

Jitian Xiao
Edith Cowan University

Chiou Peng Lam
Edith Cowan University

Huaizhong Li
Edith Cowan University

A Bipartite Graph Approach to Generate Optimal Test Sequences for Protocol Conformance Testing using the Wp-method

Jun Wang [1]
School of Computer Science and Computer Engineering, Wenzhou University China
j.wang@ecu.edu.au

Jitian Xiao
School of Computer and Information Science, Edith Cowan University Australia
j.xiao@ecu.edu.au

Chiou Peng Lam
School of Computer and Information Science, Edith Cowan University Australia
c.lam@ecu.edu.au

Huaizhong Li
School of Computer and Information Science, Edith Cowan University Australia
h.li@ecu.edu.au

Abstract

Conformance testing using test sequences is used to ensure that a protocol implementation conforms to its specification. A commonly used technique to generate test sequences for specifications described by the finite state machines is the Wp-method with the Reset technique, which frequently results in long test sequences. In this paper, we propose a bipartite graph approach to generate optimal test sequences for protocol conformance testing. Our approach significantly reduces the length of the test sequences required for conformance testing while maintaining the same fault detection capability.

1. Introduction

Many systems can be modeled using the finite state machines (FSMs). For examples, the communication protocols and some control systems can be easily interpreted as state machines [1][12]. Based on the specifications, implementations are realized to satisfy these specifications.

Protocols are a set of rule to represent the potential interactions in the system [1]. To improve the reliability and to ensure the quality of the implemented systems, it is necessary to test the protocols of the systems to ensure that they conform to the required system specifications.

Protocol conformance testing generally involves applying a sequence of inputs, which is generated from the specification, to the implementation and then verifies whether the expected sequence of outputs is obtained [1][4]. As the implementation is a black box for testing purpose, the protocol conformance testing problem is difficult to be tackled [1]. One of the most important issues in protocol conformance testing is how to generate the sequence of inputs, called the test sequence, in an efficient and effective way to achieve the required fault detection coverage.

Many methods have been proposed to tackled the protocol conformance testing problem, for examples, [1][3][4][12] and the references therein. Among the proposed methods, the W-method [5] and the improved Wp-method [7] attract much attention. It has been shown that unlike other methods, W-method and Wp-method can be applied to all protocols, and can guarantee the detection of any output and transfer faults under certain conditions [5][7].

Unfortunately, the length of the test sequences generated using the W-method and the Wp-method are usually the longest amongst all methods. Furthermore, it is commonly assumed that the implementation under test has reliable reset functions which is difficult to be realized for some systems. If an implementation doesn't have reliable reset, alternative reset technique can be used to home the test sequence [7]. The reset technique inevitably brings in additional overhead into the already very long test sequences.

Attempts have been made to optimize test sequences generated using the W-method and the Wp-method. For example, a technique involving the use of the Rural China Postman problem is proposed in [4] to optimize test sequences generated using the Wp-method for protocol conformance testing. It has been shown in [4] that the test sequences can be shortened significantly.

This paper proposes a bipartite graph approach to generate optimal test sequences for protocol conformance testing. The approach proposed in this paper is an extension to the results reported in [4]. We

[1] This author was a visiting fellow at Edith Cowan University when the paper was written.

Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05)
0-7695-2465-6/05 $20.00 © 2005 IEEE

show that the test sequences can be further optimized while maintaining the same fault detection capability.

This paper is organized as follows. Section 2 briefly describes the preliminaries on FSM. Section 3 reviews the related work. Section 4 presents a bipartite graph approach to optimal test sequence generation. Some discussions about the proposed method are detailed in Section 5. Finally, the conclusion is found in Section 6.

2. Preliminaries

Definition 2.1 [12]: A deterministic finite state machine (FSM) M is a quintuple (S, X, Y, δ, λ) where S is the finite set of states which includes the special state s_0 called the *initial state,* X is the finite set of inputs, Y is the finite set of outputs which includes "null", δ: S×X →S is the transfer function, λ: S×X →Y is the output function.

If functions δ and λ are defined for all $(s_i, x_j) \in$ S×X, the FSM M is called *completely specified*. It is always possible to render a FSM completely specified [3]. Therefore, without loss of generality, we assume that the considered FSMs are completely specified.

The transfer function δ and the output function λ can be combined to form a transition relation t. For $(s_i, x_j) \in$ S×X, if $\delta(s_i, x_j)= s_k$ and $\lambda(s_i, x_j)= y_h$, a transition $t(s_i, x_j, s_k, y_h)$ in the transition set T is defined as t: $(s_i, x_j) \rightarrow (s_k, y_h)$. If M is completely specified, |S|=*n*, and |X|=*m*, then |T|=*n*×*m*.

Definition 2.2 [1]: A labeled digraph G is denoted as G = (V, E) where V is a set of vertices and E is a set of labeled directed edges which link the vertices. An edge e which starts from v_i and ends at v_j with distinct label *l* is denoted by $e(v_i, v_j, l)$. The number of edges which start from v_i is called the *Outdegree* of v_i. The number of edges which end at v_i is called the *Indegree* of v_i.

If it is possible to partition the V set into two disjoint sets V` and V`` such that every edge of G connects a vertex in V` to a vertex in V``, G is said to be bipartite [8].

If V=S, and E includes an edge $e(s_i, s_k, x_j/y_h)$ if and only if T includes transition $t(s_i, x_j, s_k, y_h)$, then a digraph G(V, E) is a *derived digraph* of the corresponding FSM M. Due to the equivalence of e and t, in the following we will not distinguish between e and t, and will subsequently denote the derived digraph of M as G(S, T).

The function δ, function λ and transition t can be extended to input sequences. For completely specified FSMs, for a given state s_i and a given input sequence $\bar{x}=\{x_1, x_2, ..., x_n\}$, there is a transition sequence: $t_1(s_i, x_1, s_{i+1}, y_1)$, $t_2(s_{i+1}, x_2, s_{i+2}, y_2)$, ..., $t_n(s_{i+n-1}, x_n, s_{i+n}, y_n)$. We denote the transition sequence as $\bar{t}(s_i, \bar{x}) = \{t_1, t_2, ..., t_n\}$. On the other hand, given a transition sequence $\bar{t}$, there is an input sequence $\bar{x}$ which corresponds to $\bar{t}$. Therefore, for simplicity and without loss of generality, we frequently use $\bar{t}$ in the place of $\bar{x}$. Similarly, we denote the output sequence $\bar{y} = \lambda(s_i, \bar{x})=\{y_1, y_2, ..., y_n\}$ for a transfer $\delta(s_i, \bar{x})=s_{i+n}$. Note that the extension of the definitions of δ and λ in this paper is slightly different from that in [12]. In the derived digraph G(S, T), there exists an adjacent edge sequence $\bar{t}(s_i, \bar{x}) =\{ (s_i, s_{i+1}, x_1/y_1), (s_{i+1}, s_{i+2}, x_2/y_2),..., (s_{i+n-1}, s_{i+n}, x_n/y_n)\}$ corresponding to the transfer from s_i to s_{i+n}. We call $\bar{t}$ a *walk* of G from s_i to s_{i+n}, while s_i is the starting state and s_{i+n} the ending state of $\bar{t}$.

An FSM is *strongly connected* if there exists a walk between any pair of two distinct states s_i and s_j in its derived digraph. In this paper, we assume that FSM is *strongly connected* which is also a standard assumption for approaches based on the Wp-method. Strongly connected FSMs have complete reachability, i.e., there is always a feasible walk between two distinct states in the derived digraph.

The following definitions are modified from [12]:

Definition 2.3: Two states s_i and s_j of an FSM M are *equivalent* if for any input sequence $\bar{x}$, their output sequences $\lambda(s_i, \bar{x}) = \lambda(s_j, \bar{x})$. Equivalence for two states in different FSMs with the same input and output sets can be similarly defined. Two FSMs are *equivalent* if and only if for every state of one FSM, there is an equivalent state in the other FSM, and vice versa.

Definition 2.4: An FSM M is *minimal* if all other FSMs which are equivalent to M have equal or more number of states than M.

An FSM M is minimal if and only if there is no equivalent state in M [9][12]. Similar to [1][4][12], we assume that FSM M and its implementation-under-test (IUT) FSM I considered in this paper are all minimal.

If a system is specified by an FSM M, and its IUT is another FSM I, the problem to determine whether I is equivalent to M is called "conformance testing" or "fault detection" problem.

3. Review of Related Work

Many approaches, such as *T, D, W, Wp* and *UIO*, have been proposed to solve the conformance testing problem (See, for examples, [1][3][4][7][12]). Among these methods, the W-method and the Wp-method attracted greater attention as they have broader applicability and higher fault coverage [4][7][12]. In the following, we briefly review the W-method and the Wp-method.

The W-method and the Wp-method are used to generate conformance test sequences. Both methods use three input sequence sets, namely, the states cover set Q, the transition cover set P and the characterization set W.

The W-method and the Wp-method are applicable if the FSM M is completely specified, strongly connected and minimal. Under this assumption, the homing sequence theory [9] guarantees that for each state s_i, there is an input sequence $\bar{x}_i$ such that $\bar{t}(s_0, \bar{x}_i)$ is a *walk* from s_0 to s_i of G. If $s_i \neq s_j$, and $\bar{y}_i = \lambda(s_0, \bar{x}_i) \neq \lambda(s_0, \bar{x}_j) = \bar{y}_j$, denote $Q = \{\bar{x}_i \mid s_i \in S\}$, we call Q the state cover set for M. Furthermore, suppose that there are k_i transitions $t_{i1}(s_i, s_{i1}, x_{i1}/y_{i1})$, $t_{i2}(s_i, s_{i2}, x_{i2}/y_{i2})$, ..., $t_{iki}(s_i, s_{iki}, x_{ik_i}/y_{ik_i})$ starting from s_i, let $P=\{\bar{x}_i . x_{ij} \mid s_i \in S, j=1,\ldots,k_i\}$, then P is the transition cover set for M. Apparently, if $|S|=n$ and $|X|=m$, $|P|=n \times m$. The characterization set W is an input sequence set such that for any pair of two states s_i and s_j in S, $\lambda(s_i, W) \neq \lambda(s_j, W)$ if $s_i \neq s_j$. Obviously, the set W can distinguish all states in S. It has been proved that W-method can detect any single transition fault, including output error and transfer error [5].

If we only need to identify a state s_i, a subset W_i of the set W is sufficient. The W_i set, which called identification set of state s_i, is constructed to have the property that for each state $s_j \neq s_i$, $\lambda(s_j, W_i) \neq \lambda(s_i, W_i)$. However, for another state $s_k \neq s_i$, it is not necessary to satisfy the relation $\lambda(s_j, W_i) \neq \lambda(s_k, W_i)$. We call W_i the identification set for state s_i.

The W-method constructs test sequences by concatenating each element of the transition cover set P to each element of the characterization set W generated using the W-method. We can represent the test sequence set for the W-method as P.W. If $|W|=w$, $|S|=n$, and $|X|=m$, then the W-method generates $n \times m \times w$ test sequences.

The Wp-method is an improved version of the W-method. The Wp-method considers the W set as a union of all W_i sets. If an element in P ends at state s_i, the Wp-method only needs to concatenate the element with those elements in W_i. Let P_i be a subset of P, and P_i includes all test transitions which ends at state s_i, the test sequence set generated using the Wp-method is $\bigcup_{i=1}^{n} P_i.W_i$. Since $|\bigcup_{i=1}^{n} P_i.W_i| << n \times m \times w$, the number of test sequences generated using the Wp-method is much smaller than that using the W-method. However, it has been shown that the two methods have the same fault detecting capability [7].

In general, a test sequence of the Wp-method can be partitioned into three parts: the prefix segment, the transition to be checked and the suffix segment. The prefix segment is an element of a state cover set Q which starts from the initial state s_0 and ends at the starting state s_i of the transition to be checked. Supposing that the transition ends at state s_j, the suffix segment of the test sequence is one of the elements of the state identification set W_j. For example, in Figure 1, the state identification set for s_4 is $W_4=\{\{T5,T6\},\{T7\}\}$.

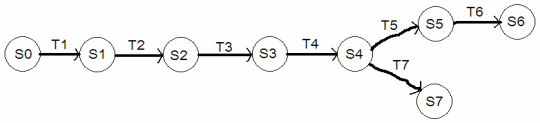

Figure 1 Test sequences for a transition

In order to detect the transition T4, two test sequences have to be generated:

{T1,T2,T3,T4,T5,T6} and {T1,T2,T3,T4,T7}.

Therefore, the prefix segment is {T1,T2,T3}, and the suffix segments are {T5,T6} and {T7} respectively. The transition to be tested together with the suffix segment is called the "*test segment*". In the Figure 1 example, {T4,T5,T6} and {T4,T7} are two test segments to detect transition T4. Note that we use the transition sequence to represent the corresponding input sequence here.

However, the length of test sequences which is generated by these two methods is generally long. Both the W-method and the Wp-method generate a set of test sequences which start from the initial state of M. Therefore, it is necessary to return to the initial state before applying each test sequence. To simplify the situation, it is frequently assumed in literature that the tackled FSM M has reset operations such that it is possible to reset to the initial state from any state immediately. This means that each of states has a reset transition from the state to the initial state.

If M does not have such reset operations, and M is strongly connected, it is necessary to construct a walk from a state back to the initial state. Such walks are called *initial sequences*. Obviously, in order to apply all of test sequences, it is necessary to return from the ending states to the initial state many times. This implies that a very long test sequence is usually constructed by linking all of test sequences and all of the initial sequences. Clearly, this implicit test sequence is not optimal for conformance testing.

Attempts have been reported to reduce the length and the redundancy of the test sequence, especially when the FSM M does not have reset operations (see

for examples, [1][3][4] and the references therein). In principle, it is recognized that it is not necessary to start all tests in the initial state. After apply a test segment, it is possible to use a so-called transfer sequence to continue the testing from the final state of the current test segment to the starting state of the next test segment immediately. This obviously avoids the redundant testing of the initial sequences and the prefix segments which are already tested before.

An optimization technique for conformance testing based on the Wp-methods is reported in [4]. The technique involves the construction of the test segments using the Wp-method, and a Rural Chinese Postman (RCP) algorithm is applied to optimally connect these test segments into a test sequence. It is shown in [4] that the constructed test sequence is much shorter than the implicit test sequence constructed by linking all of test sequences and all of the initial sequences. The technique used in [4] is briefly introduced below.

The algorithm in [4] consists of 4 steps, namely, 1. Generate the test segment set Cw from the Wp sets; 2. Construct a W digraph G'(V', E'). First Copy the transition digraph G(V, E) of the FSM into G'(V', E'); then, for each test segment which starts from state J and ends at state K, a corresponding bold edge is added to graph G'(V', E'). The edge cost is assigned as the length of the test segment; 3. Graph Modification. Check if all bold edges of G'(V', E') are weakly connected using an algorithm reported in [1]. If those bold edges are not weakly connected then a minimum set of non-bold edges is augmented using the algorithm of [6] to render the augmented edges weakly connected; 4. Test Sequence Generation. Use the RCP algorithm proposed in [1] to find an RCP tour which traverse the bold edge of G'(V, E').

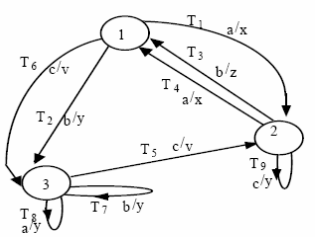

Figure 2 The transition digraph G(V, E) of M [4]

The example shown below is taken from [4]. Consider an FSM M represented by the transition digraph G(V, E) in Figure 2. All the assumptions used in [4] are adopted here. Note that it was declared that only transitions T1, T2, …, T6 needed to be checked as these transitions represented the main behavior of the protocol [4]. For comparison purpose, we also adopt this declaration.

A set of test segments for M is obtained in [4]. The details of the test segments are shown in Table 1. It is reported in [4] that an optimal test sequence for the FSM in Figure 2 has the total length of 23.

Table 1 A set of test segments for the FSM M(Cw)

Starting State	Test Segments	Ending State
1	Check(T1)=[T1, T3]	1
1	Check(T2)=[T2, T8]	3
2	Check(T3)^1=[T3, T1]	2
2	Check(T3)^2=[T3, T2]	3
2	Check(T4)^1=[T4, T1]	2
2	Check(T4)^2=[T4, T2]	3
3	Check(T5)=[T5, T3]	1
1	Check(T6)=[T6, T8]	3

4. An Optimized Test Sequence Generation Technique based on the Wp Method

In this section, we describe our approach used to further improve the results report in [4].

Many of these segments are overlapped in the set of test segments Cw. These overlapped test segments can be merged to a longer test segment to reduce the number of segments. Let set Cm consists of these merged test sequences. Furthermore, the final states of some test segments in the Cm set are the same as the starting states of some other test segments. These segments can be linked to form one test segment which further reduces the number of test segments. Let set Cl denote these linked test sequences. Afterwards, we use these test segments in Cl to form the shortest test sequence.

There are many methods to form the shortest test sequence from the test segments. For example, RCP method [1] can be applied. In this paper, we propose a bipartite graph method to tackle the addressed problem. Detailed discussion about the method will be addressed in section 5.

The proposed method consists of 4 steps outline below:

Step 1: Generate the test segment set Cw from the Wp testing segment set. This step is as same as the first step in [4];
Step 2: Merge the segments in Cw to form the set Cm;
Step 3: Link the segments in Cm to form the set Cl;
Step 4: Generate a weighted bipartite digraph G, and find a shortest test sequence.

First of all, we present the algorithm shown in Table 2 to merge the overlapped test segments in Cw:

Table 2 The merge algorithm

Merge Algorithm

Step 1: Let Cm be an empty set.

Step 2: Create a weighted digraph G(Cw , Ew). For each vertex c_i and c_j, suppose that c_j includes m transitions and the anterior parts of k transitions for c_j overlap with the posterior parts of the c_i transitions. If m-k=0, the redundant c_j is removed from Cw. Otherwise let w_{ij}=k/(m-k). If w_{ij}>0 (which means k>0), we draw a arc e_{ij} from c_i to c_j with weight w_{ij}. The Ew set consists of these arcs;

Step 3: Loop while Cw is not empty

Step 3.1: Move each of discrete vertex (which has no arc) to Cm;

Step 3.2: If Cw is not empty,

for each disjoint subgraph of G(Cw , Ew)
select an arc e_{ij} which has the maximal weight, merge the vertex c_j to c_i, then remove c_j from Cw, delete all arcs which link to cj and redraw the weight arcs between ci and other vertexes.

else
break the loop

If there exist test segments in Cm whose final states are the same as the starting states of some other test segments, these test segments can be linked together to reduce the number of test segments further. The link algorithm is shown in Table 3:

Table 3 The link algorithm

Link Algorithm

Step 1: Let Cl be an empty set.

Step 2: loop while Cm is not empty

for each element c_i of Cm,
if there is another element c_j whose starting state is the same as the ending state of c_i,
link c_j to c_i, then delete c_j from Cm;
elseif there is another element c_k whose ending state is the same as the starting state of c_i,
link c_k to c_i, then delete c_k from Cm;
else
move c_i to Cl.

Next, we describe the algorithm to combine the test segments in Cl to form the shortest test sequence.

The first step is to construct a weighted bipartite digraph B(V, E). The vertex set V of the digraph B consists of two subsets V` and V``, V=V` ∪ V``. The V` subset consists of the starting states of test segments in Cl and the V`` subset consists of the ending states of test segments in Cl. Similarly, the E set consists of two types of edges. For every test segment c_i in Cl, if the starting state and the ending state of c_i are s_{is} and s_{ie}, we draw an arc from s_{is} to s_{ie}, label the edge by *"ci"*, and assign the weight 0 to the edge. These edges, which all go from V` to V``, form the first edge subset E` in E. The second edge subset E`` can be constructed in a similar way, it consists of the edges which all go from V`` to V`. For every pair of state ($s_i$, $s_j$), $s_i \in V``$, $s_j \in V`$ and $s_i \neq s_j$, we draw an arc e_{ij} from s_i to s_j. The arc's weight w_{ij} is the number of arcs which are the shortest path from s_i to s_j in the derived digraph G(S, T) of FSM M.

The second step is to solve the RCP problem to find the shortest tour which include all edges in E`. One of the algorithms to solve the RCP problem is to represent it as an Integer Linear Programming (ILP) problem. Let a_{ij} represent the number of times that arc e_{ij} is used in the route, the objective function is

$$\min \ w=\sum_{i\neq j} a_{ij} * w_{ij}$$

Let I_i represents the Indegree of $s_i \in V``$, O_j represents the Outdegree of $s_j \in V`$, then the constraints for the ILP problem are

$$\begin{cases} \sum_{allj\neq i} a_{ij} = I_i \quad (\text{for each } s_i \in V``) \\ \sum_{alli\neq j} a_{ij} = O_j \quad (\text{for each } s_j \in V`) \end{cases}$$

Many algorithms can be applied to solve the ILP problem [14], hence the details are omitted here.

Now we present two examples to demonstrate the proposed method. The first example is taken from [4]. The FSM M is represented in Figure 1 and the test segment set Cw is shown in Table 1, namely

Cw={c1=Check(T1)=[T_1,T_3], c2=Check(T2)=[T_2, T_8], c3=Check(T3)^1 =[T_3,T_1], c4=Check(T3)^2 =[T_3,T_2], c5=Check(T4)^1 =[T_4,T_1], c6=Check(T4)^2 =[T_4,T_2], c7=Check(T5) =[T_5,T_3], c8=Check(T6) =[T_6,T_8] }

Due to space limitation, the detailed steps are omitted. Applying the proposed algorithms, we obtain a weighted digraph B(V, E) shown in Figure 3.

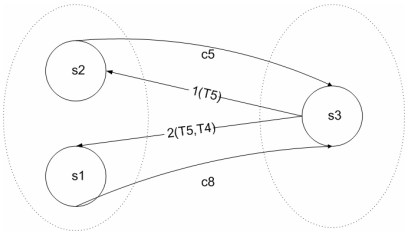

Figure 3 The digraph B(V, E)

The path in Figure 3 which has minimum weight is 1 (T5). Therefore, the short test sequence is

T6, T8, T5, **T4,T1, T3, T1, T4, T2, T8, T5, T3, T2** which has the length 13 comparing with the length 23 obtained in [4].

The second example is more complicated. The FSM is given by a digraph represented in Table 4 where s_1 is the initial state.

Table 4 The FSM for the second example

Transition	Starting State	Input/Output	Ending State
t1	1	a/0	2
t2	1	b/0	4
t3	2	a/0	5
t4	2	b/1	6
t5	3	a/1	6
t6	3	b/0	1
t7	4	a/1	2
t8	4	b/1	3
t9	5	a/0	6
t10	5	b/1	1
t11	6	a/1	4
t12	6	b/0	6

The test segment set Cw is
Cw={c1 = Check(t1) = [t1,t3,t9,t11], c2 = Check(t2) = [t2,t7,t3], c3 = Check(t3) = [t3,t10,t1], c4 = Check(t4) = [t4,t12,t11,t7], c5 = Check(t5) = [t5,t12,t11,t7], c6 = Check(t6) = [t6,t2,t8], c7 = Check(t7) = [t7,t3,t9,t11], c8 = Check(t8) = [t8, t5,t12], c9 = Check(t9) = [t9,t12,t11,t7], c10 = Check(t10) = [t10,t2,t8], c11 = Check(t11) = [t11,t7,t3], c12 = Check(t12) = [t12,t12,t11,t7]}

The created weighted digraph B(V, E) is shown in Figure 4. The derivation details are also omitted.

Finding the shortest sequence in B leads to solving the following ILP:

$$\min \quad 2x_{21} + x_{25} + x_{26} + 2x_{41} + 2x_{45} + 2x_{46}$$

$$\begin{cases} x_{21} + x_{41} = 1 \\ x_{25} + x_{45} = 2 \\ x_{26} + x_{46} = 1 \\ x_{21} + x_{25} + x_{26} = 2 \\ x_{41} + x_{45} + x_{46} = 2 \end{cases}$$

An optimal solution for the above ILP is:

$$x_{21} = 0, x_{25} = 2, x_{26} = 0, x_{41} = 1, x_{45} = 0, x_{46} = 1$$

Therefore a shortest path starting from s_1 is:

c2→e46→c12→e25→c10→e25→c9

The walk corresponds to the above the path is:

t2,t7,t3,t9,t11,t8,t5,**t12,t12,t11,t7**,t3,**t10,t2,t8,t6,t2,t8, t5,t12,t11,t7**,t3,**t9,t12,t11,t7,t4,t12,t11,t7,t3,t10,t1,t3, t9,t11**

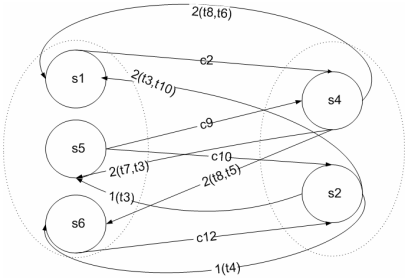

Figure 4 B(V,E) for the second example

Note that only four additional transitions are added in to form the optimal test sequence. The corresponding input sequence is:

b a a a a b a **b b a a** a **b b b b b b a b a a** a **a b a a b b a a a b a a a a**

5. Discussions

In the section, we present discussions on the various parts of the proposed method.

5.1 On the merge algorithm

The merge algorithm is the most important one in the proposed method. There are many test segments in a Wp-set which can be merged into one test sequence. The merge algorithm reduces the number of test segments and hence shortens the length of final test sequence. However, it needs to show that the merged test segments have the same fault detecting capability.

In the Wp-method, an error is detected if there are some unexpected outputs corresponding to inputs in a test segment. By finding which test segment the error is detected, we can determine the nature and the location of the error. The test segments in our method after the merge and link have explicit correspondence relation to those segments in the Wp-mthod. To show this, we only need to show that any unexpected output which is detected by a Wp-method test segment will also be detected by the merged and linked test segments generated using our method.

Suppose that the test segment in the Wp-method which detects an error is

$$x_0/y_0, x_1/y_1, x_2/y_2, \ldots, x_k/y_k$$

and assume that the test segment starts at state s_i. Then the transition segment corresponding to the test segment is

$$t_0(s_i, s_{i+1}, x_0/y_0), t_1(s_{i+1},s_{i+2}, x_1/y_1), t_2(s_{i+2}, s_{i+3},x_2/y_2), \ldots, t_k(s_{i+k}, s_{i+k+1}, x_k/y_k)$$

If in the merging process, this test segment is not merged with other test segments, or it becomes the

anterior part of a merged test segment, certainly its error detecting capability is not affected by the merge.

Now we consider the case that the test segment is merged as the posterior part of the merged test segment. Suppose that the anterior part of the merged test segment is

$$x'_0/y'_0, x'_1/y'_1, \ldots, x'_m/y'_m$$

its starting state is s_j, and its corresponding transition segment is

$$t'_0(s_j, s_{j+1}, x'_0/y'_0), t'_1(s_{j+1}, s_{j+2}, x'_1/y'_1), \ldots,$$

$$t'_m(s_{j+m}, s_{j+m+1}, x'_m/y'_m)$$

The whole merged test segment is

$$x'_0/y'_0, x'_1/y'_1, \ldots, x'_m/y'_m, x_{p+1}/y_{p+1}, \ldots, x_r/y_r, \ldots, x_k/y_k$$

According to our merge algorithm, merge action implies the facts that $0 \le p < m$, $p < k$, the transition $t'_{m-p}(s_{j+m-p}, s_{j+m-p+1}, x'_{m-p}/y'_{m-p})$ is the same as the transition $t_0(s_i, s_{i+1}, x_0/y_0)$, and the transition segment from $t'_{m-p}(s_{j+m-p}, s_{j+m-p+1}, x'_{m-p}/y'_{m-p})$ to $t'_m(s_{j+m}, s_{j+m+1}, x'_m/y'_m)$ is the same as the transition segment from $t_0(s_i, s_{i+1}, x_0/y_0)$ to $t_p(s_{i+p}, s_{i+p+1}, x_p/y_p)$. Therefore, if there is an unexpected output in the anterior part of the segment, the fault will be already detected before the merged part is reached. Otherwise, if no error occurs while we apply the test segment $x'_0/y'_0, x'_1/y'_1, x'_2/y'_2, \ldots\ldots, x'_m/y'_m$ generated using the Wp-method, there will be no error occur in the anterior part of the merged test segment generated using our method. This implies that when the merged test segment is executed, the state s_i can be reached by applying the anterior part of the segment $x'_0/y'_0, x'_1/y'_1, x'_2/y'_2, \ldots\ldots, x'_{m-p-1}/y'_{m-p-1}$, afterwards, the remaining part of the merged test segment is applied. It is interesting to observe that applying the remaining part of the merged test segment is the same as applying the test segment $x_0/y_0, x_1/y_1, x_2/y_2, \ldots\ldots, x_k/y_k$ in the Wp-method, that is, we start at the same state s_i and apply the same test segment. Clearly, the fault detecting capability should be the same as well.

5.2 On the heuristic function w(i, j)

In the merge algorithm, the heuristic function w(i, j) = k/(m-k) is employed to calculate the total weight of the edges in the weighted digraph G(Cw , Ew) and to prioritize the merging of the test segments. Using the ratio of the length of the overlapped part to the length of the remaining part, it is possible to further shorten the length of the merged test segment.

When some edges have equal weight, one of the edges can be arbitrarily selected. However, the different selection may result in a different starting state of a test segment and may affect the succeeding steps in our method. Therefore, the heuristic function used in our method can only achieve local optimal. The question of how to construct a heuristic function which achieves global optimal remains an open problem.

5.3 On the link algorithm

In the link algorithm, we can select different test segments to link. The different selections result in different sets of Cl. Fortunately, these different sets of Cl are equivalent in the sense that they include the same number of test segments which correspond to the same starting and ending states. Therefore, these sets of Cl produce the same bipartite digraph B(V, E).

There are two classes of test segments in Cl. The first class groups these test segments where the starting states are the same as the ending states. The other class contains the remaining test segments. Suppose that there are k test segments in the first class, then the number of vertices of V is no more than $n+k$. This is due to fact that the k starting states of the test segments in the first class belong to V`, the corresponding k ending states which are the same as the k starting states belong to V``, and the other states, at most $n-k$, belong to V` or V`` respectively. Consequently, the number of edges e_{ij} which go from V`` to V` is no more than $((n+k)/2)^2 - k$ which is an upper bound of the number of the decision variables for the ILP problem mentioned above. In fact, suppose that $|V``|=p$ and $|V`|=q$, the number of the decision variables is $pq-k$. This is correct because the arcs are drawn to link pairs of different states from V`` to V`, and there exist $k$ same states in V`` and V`, therefore the number of edges going from V`` to V` is $k(q-1)+(p-k)q = pq-k$.

Note that $p+q \le n+k$. According to the well known mathematic average inequality $pq \le ((p+q)/2)^2$, it is easy to conclude that $pq-k \le ((n+k)/2)^2 - k$.

5.4 On the shortest path searching algorithm

Many algorithms exist to find an optimal solution of the mentioned ILP. Furthermore, approximation algorithms with complexity of lower order polynomial time can be applied to find a good (which may not be optimal) solution to the ILP [14].

For the completeness of our method, we briefly propose an approximation algorithm here. Namely, we

sort the arcs e_{ij} by their weights and arrange them in the order from the highest weight to the lowest one. Then we take turns to evaluate each of the corresponding decision variables a_{ij} to get a smallest nonnegative integer which satisfies the constraints of the ILP. This algorithm is easy to be used, and it has $O(n^2)$ complexity.

Before we conclude this paper, we would like to emphasize the following points:

1. The optimal solution in [4] relies on the construction of the RCP tour. It is well-known that the RCP is NP-hard. Although heuristic algorithms were proposed in [1], it was noted that the implementation for the RCP was not reported in literature [13]. Converting a RCP into an ILP, as shown in this paper, is a promising way to solve the RCP as many algorithms are available to solve the ILP problems. Unlike the RCP, approximation algorithms like the one proposed in this paper can be used to solve the ILP problems efficiently [14];
2. If the original test segments in [4] do not form a weakly connected sub-graph, which is frequently the case for complicated systems, additional edges have to be augmented such that the augmented edges and the original bold edges form a weakly connected sub-graph, then the heuristic algorithms for the RCP can be used to find an RCP tour. Unfortunately, optimal is achieved for the augmented sub-graph only which means that inevitably all augmented edges have to be adopted into the final test sequence. Therefore, the test sequence produced in [4] may not be optimal;
3. In our approach, we propose several systematic algorithms and steps to merge and link the test segments for a FSM which lead to the construction of a weighted bipartite digraph. The optimal test sequence is obtained by finding the shortest tour which includes all edges in E` of the bipartite diagraph. As the constructed bipartite digraph is much smaller in comparison with any sub-graph created for the same FSM using the approach reported in [4], the complexity involved in solving the shortest tour problem is reduced. Furthermore, our approach avoids the excessive use of augmented edges, and therefore, leads to the generation of shorter test sequences in general.

One final note is that we need to generate a test sequence which starts from the initial state s_0 of FSM M. If $s_0 \in V$, we can simply start the test sequence from s_0. Otherwise we add s_0 to V`` and draw arcs from $s_0$ to every state in V` and count the weights of these arcs which is similar to what we do on other arcs from V`` to V`. Then we select the arc with the minimal weight as a prefix of the test sequence. The rest follows straightforwardly.

6. Conclusion

This paper describes a bipartite graph approach to optimally generate test sequences for protocol conformance testing using the Wp-method. The approach proposes systematic steps and algorithms for the generation of highly optimized test sequences. We show that the test sequences generated using the proposed method are optimal in comparison with the previously reported results, while still maintaining the same fault detection capability.

References

[1] A.V. Aho, A. T. Dahbura, D. Lee, and M. U. Uyar, "An optimization technique for protocol conformance test generation based on UIO sequences and rural Chinese postman tours", *IEEE Transactions on Communications,* 39(11): 1604-1615, 1991.

[2] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, Mass.: Addison-Wesley, 1974.

[3] J. Chen, R.M. Hierons, H. Ural, and H. Yenigun, "Eliminating Redundant Tests in a Checking Sequence", Proc. TESTCOM 2005, LNCS 3502, pp. 146-158, 2005.

[4] W.H. Chen, "An Optimization Technique for Protocol Conformance Testing Based on the Wp Method", Int. Journal of Applied Science and Engineering, 1(1):pp. 45-54, 2003.

[5] T.S. Chow, "Testing Design Modelled by Finite-State Machines", IEEE Trans. Software Eng. 4(3), 1978.

[6] K.E. Eswardan and R.E.Tarjan, "Augmentation Problems". *SIAM J. Computing,* 5(4): 653-665, 1976.

[7] S. Fujiwara, G.v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi, "Test selection based on finite state models", *IEEE Trans. on Software Eng.*, 17: 591-603, 1991.

[8] A. Gibbons. *Algorithmic graph theory*, Cambridge University Press, 1985

[9] Z. Kohavi, *Switching and Finite Automata Theory,* 2nd Ed. McGraw-Hill, 1978.

[10] J.K. Lenstra and A.H.G. Rinnooy Kan, "On general routing problems," *Networks*, 6: 273-280, 1976.

[11] D. Lee and M. Yannakakis, "Testing finite state machines: state identification and verification", *IEEE Trans. on Computers*, 43(3): 306-320, 1994.

[12] D. Lee, and M. Yannakakis, "Principles and methods of testing finite state machines", *Proceedings of IEEE,* 84(8): 1090-1123, 1996.

[13] H. Thimbleby, "The directed Chinese Postman Problem", *Software: Pract. & Exp.*, 33:1081-1096, 2003.

[14] W.L. Winston, *Operations Research Applications and Algorithms*, Duxbury, Belmont California, USA, 1993.